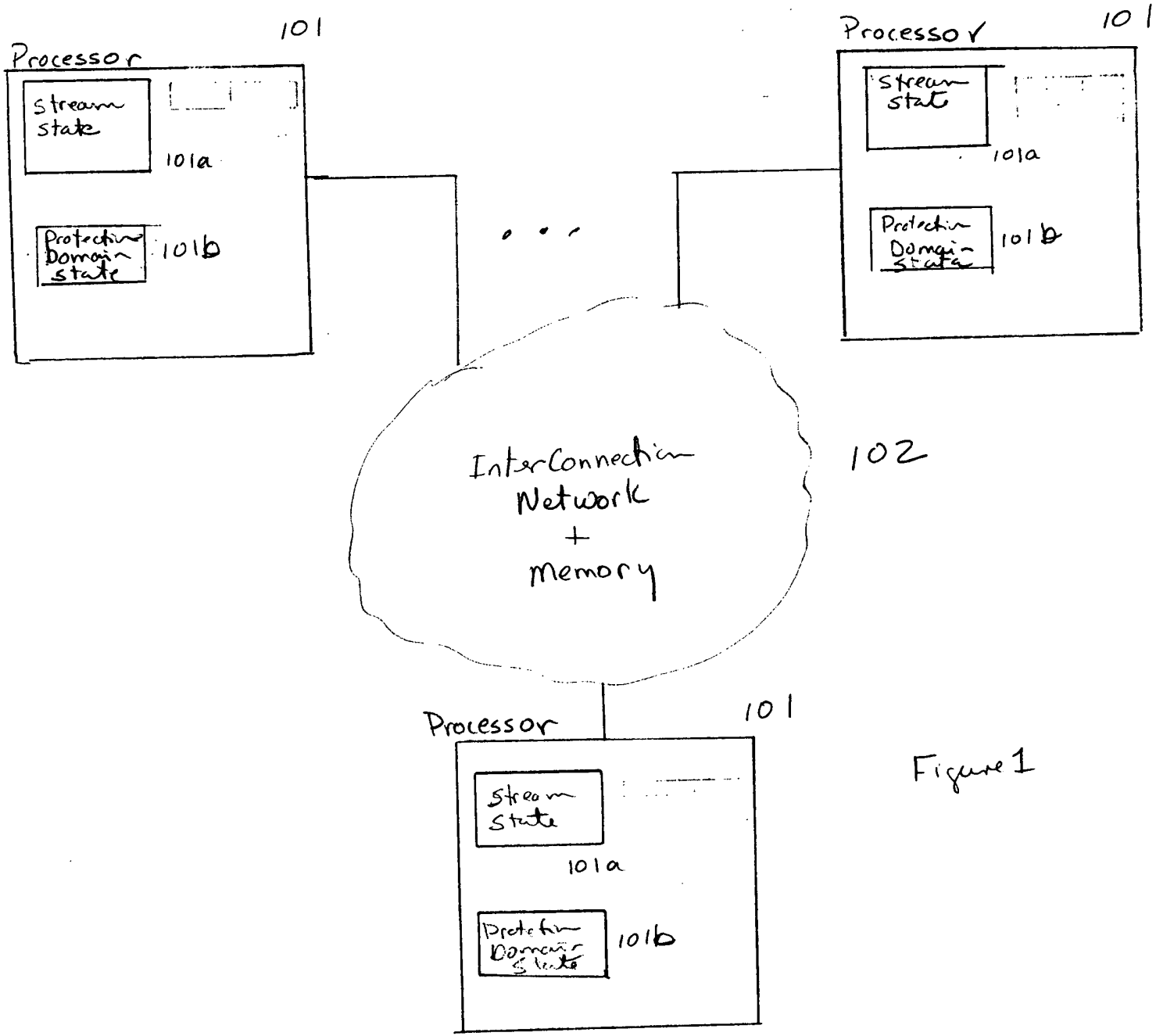

Processor
101
Stream state
101a
Protection Domain state
101b
Processor
101
Stream state
101a
Protection Domain state
101b
InterConnection Network + Memory
102
Processor
101
Stream State
101a
Protection Domain State
101b

Figure 1

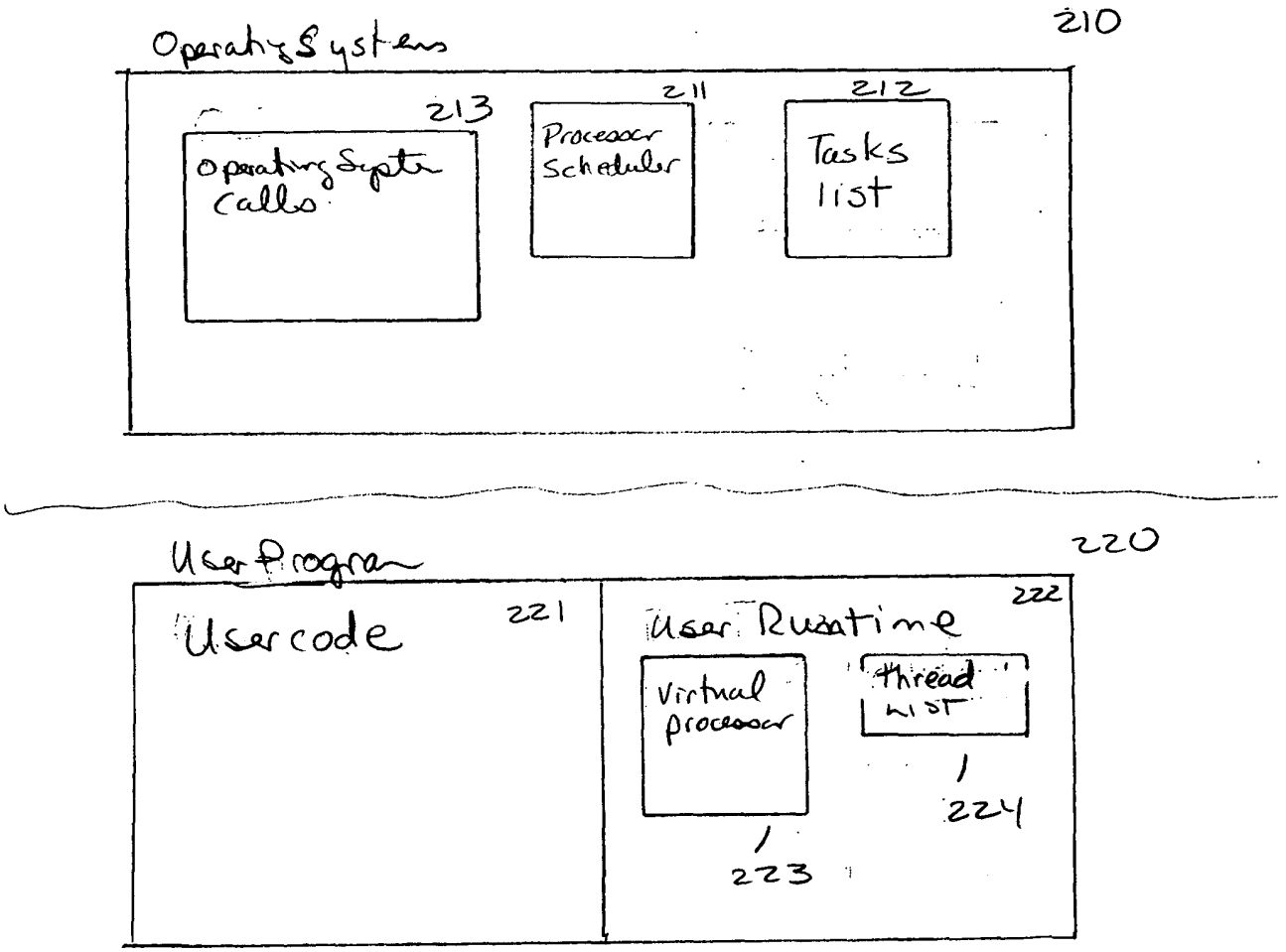

Operating System
210
213
Operating System Calls
211
Processor Scheduler
212
Tasks list
User Program
220
User code
221
User Runtime
222
Virtual processor
223
thread list
224

Figure 2

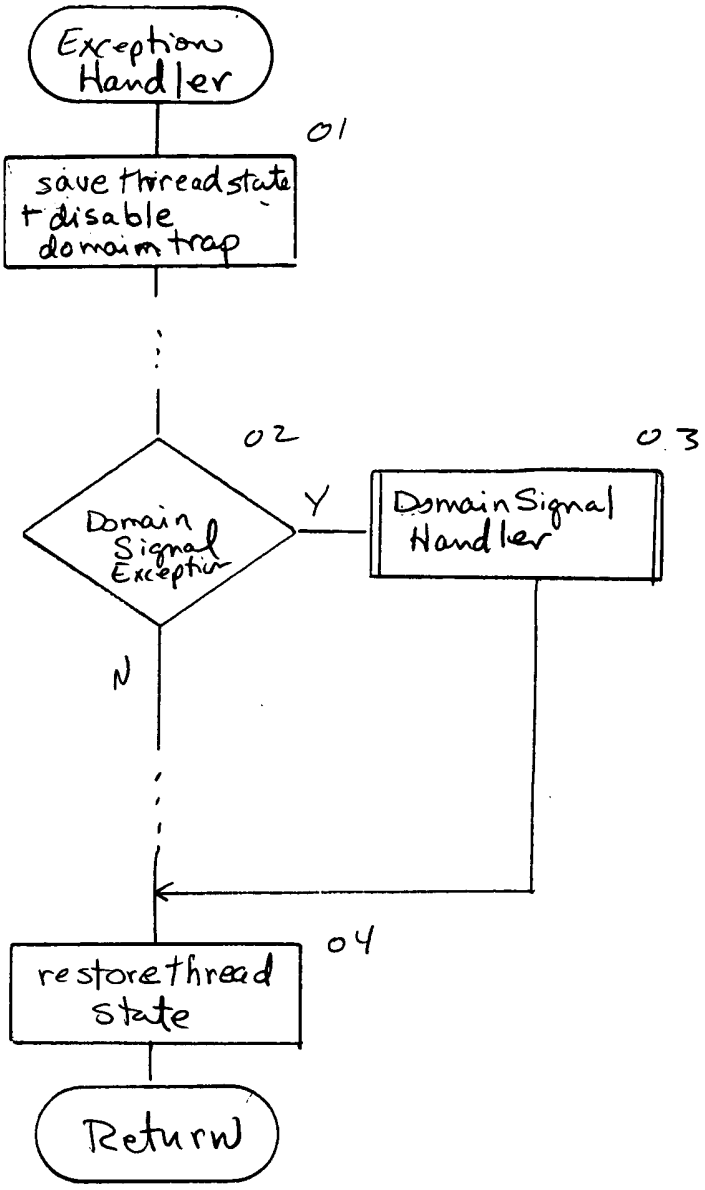

Exception Handler
01
save thread state + disable domain trap
02
Domain Signal Exception
Y
03
Domain Signal Handler
N
04
restore thread state
Return

Fig 3

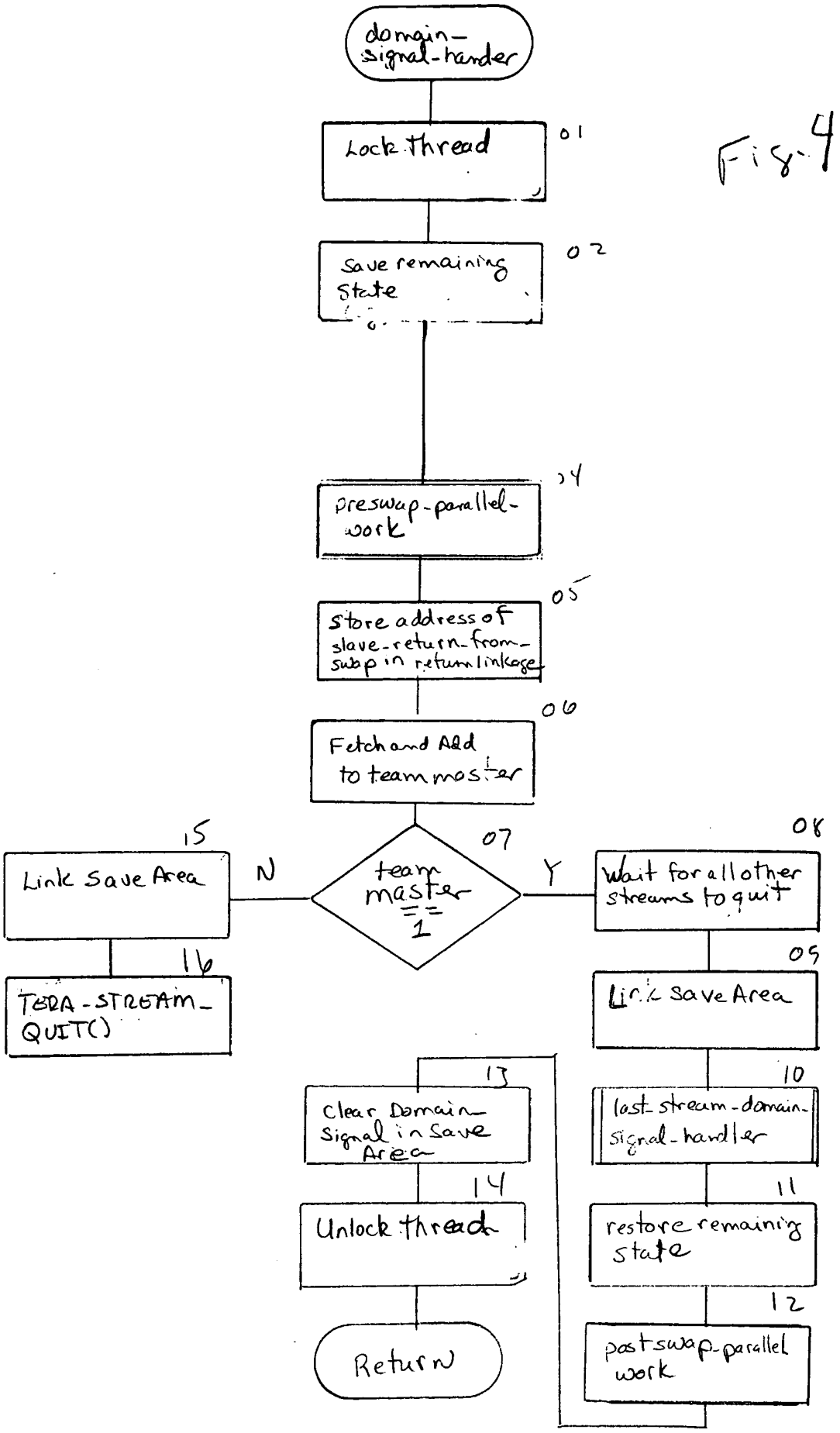

domain-signal-handler
Lock thread
01
Save remaining state
02
preswap-parallel-work
04
Store address of slave-return-from-subp in return linkage
05
Fetch and Add to team master
06
team master == 1
07
N
Y
Link Save Area
15
TERA-STREAM-QUIT()
16
wait for all other streams to quit
08
Link Save Area
09
last-stream-domain-signal-handler
10
restore remaining state
11
postswap-parallel work
12
Clear Domain Signal in Save Area
13
Unlock thread
14
Return

Fig. 4

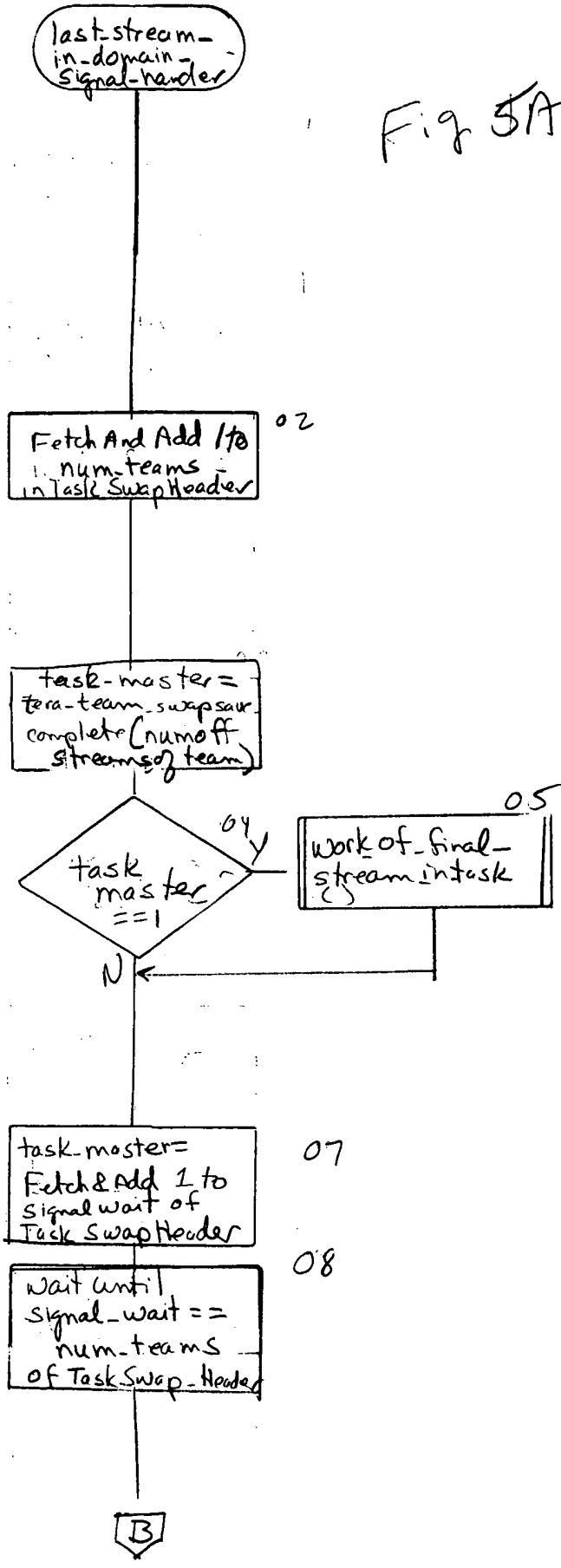

last_stream_in_domain_signal_handler
Fetch And Add 1 to num_teams in Task SwapHeader
02
task_master = tera_team_swapsaur complete (numoff streams of team)
task master == 1
04
Y
N
05
work_of_final_stream_in_task ( )
task_master= Fetch & Add 1 to signal wait of Task SwapHeader
07
Wait until signal_wait == num_teams of TaskSwap_Header
08
B

Fig 5A

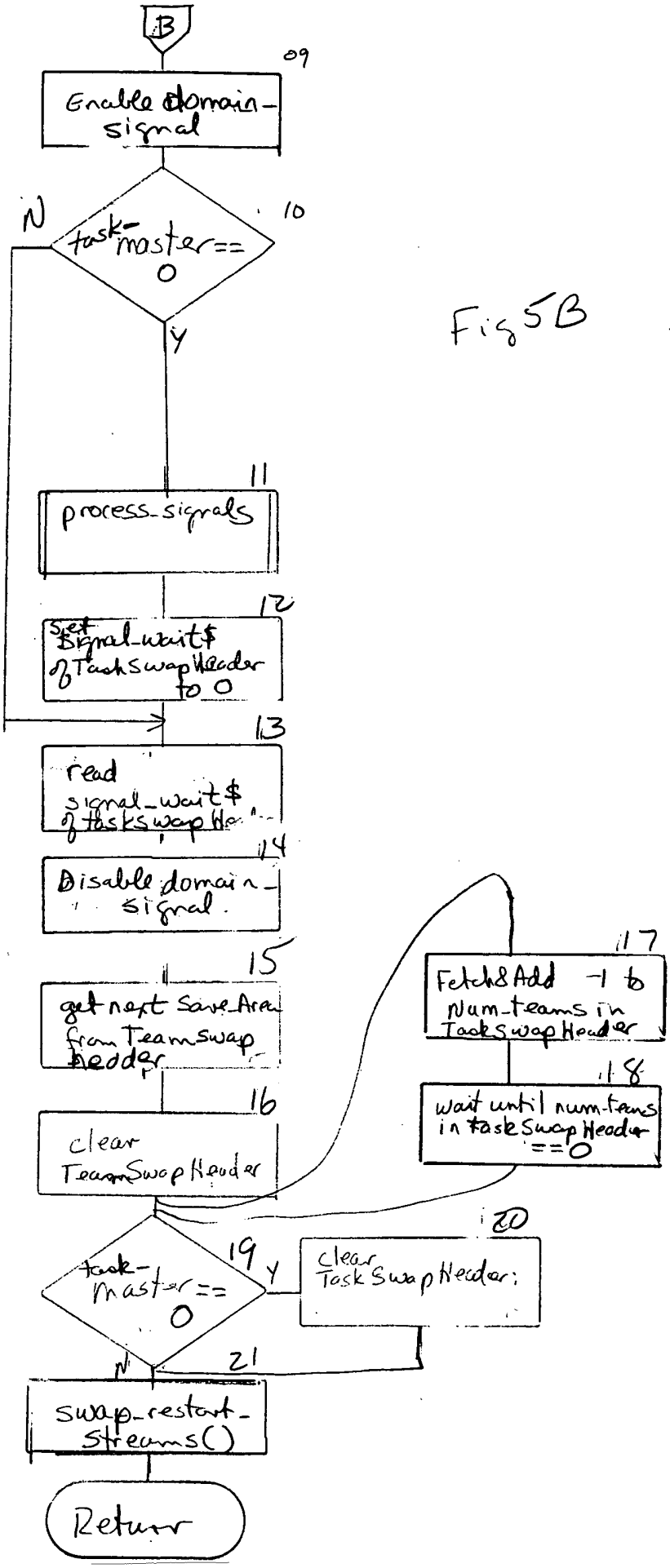

B
09
Enable domain-signal
N
task-master== 0
10
Y
11
process_signals
12
Set Signal_wait$ of TaskSwapHeader to 0
13
read signal_wait$ of taskswapHe
14
Disable domain-signal
15
get next SaveArea from TeamSwap Header
16
clear TeamSwapHeader
17
Fetch&Add -1 to Num_teams in TaskSwapHeader
18
wait until num_teams in taskSwapHeader == 0
19
task-master== 0
Y
20
clear TaskSwapHeader;
N
21
swap_restart_streams()
Return

Fig 5B

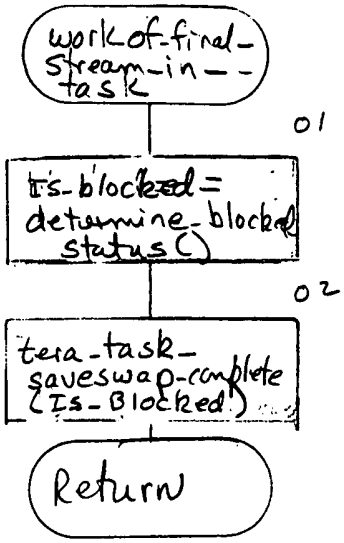

work_of_final_stream_in_task
01
Is_blocked = determine_blocked_status()
02
tera_task_saveswap_complete (Is_Blocked)
Return

Fig. 6

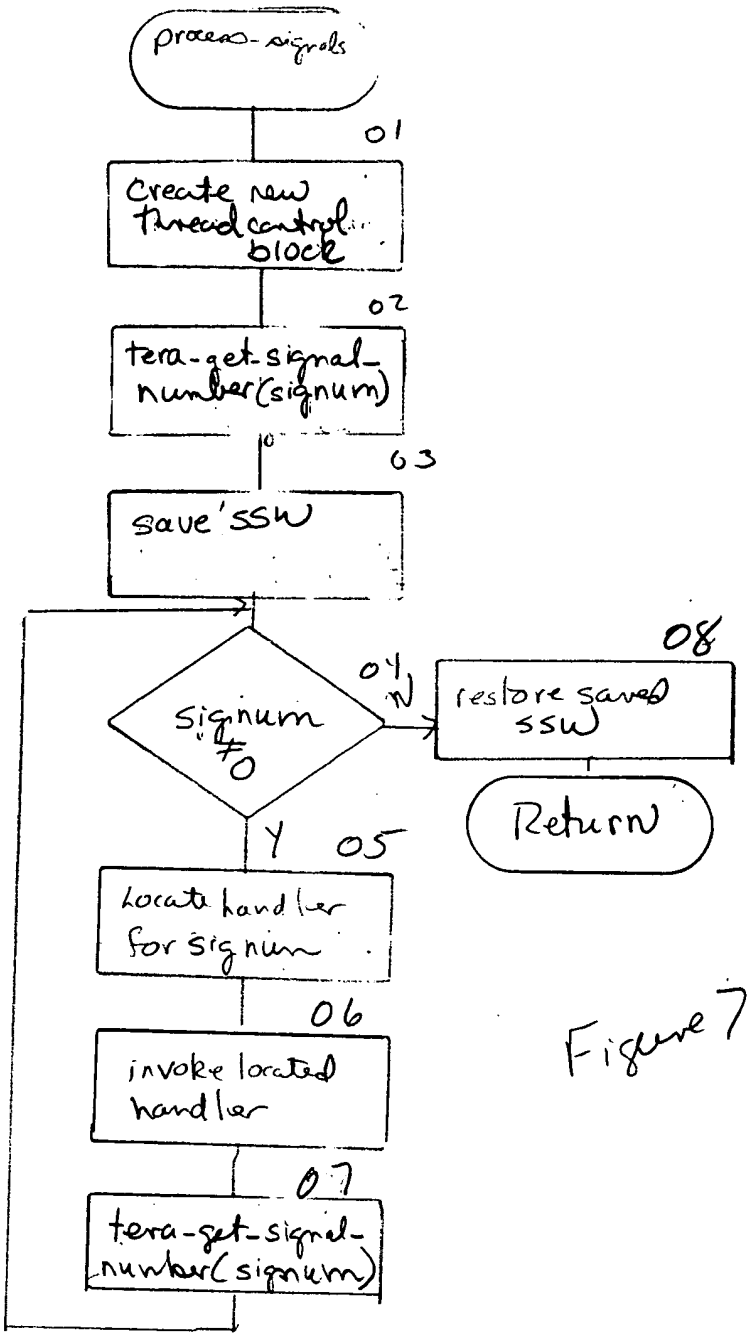

process-signals
01
Create new thread control block
02
tera-get-signal-number(signum)
03
save SSW
04
signum ≠ 0
N
08
restore saved SSW
Return
Y
05
Locate handler for signum
06
invoke located handler
07
tera-get-signal-number(signum)

Figure 7

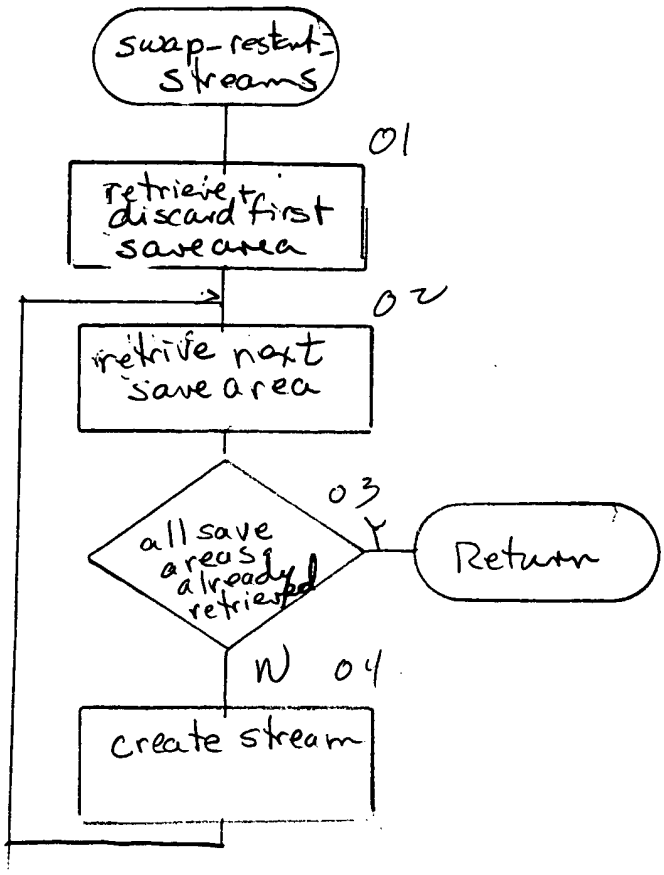

swap-restart-streams
01
retrieve & discard first save area
02
retrive next save area
all save areas already retrieved
03
Return
N
04
create stream

Fig 8

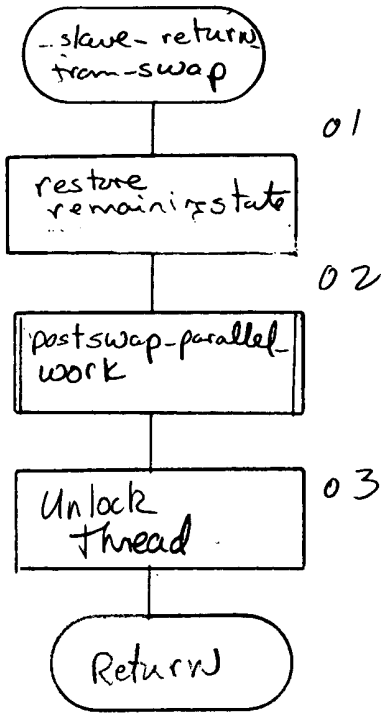
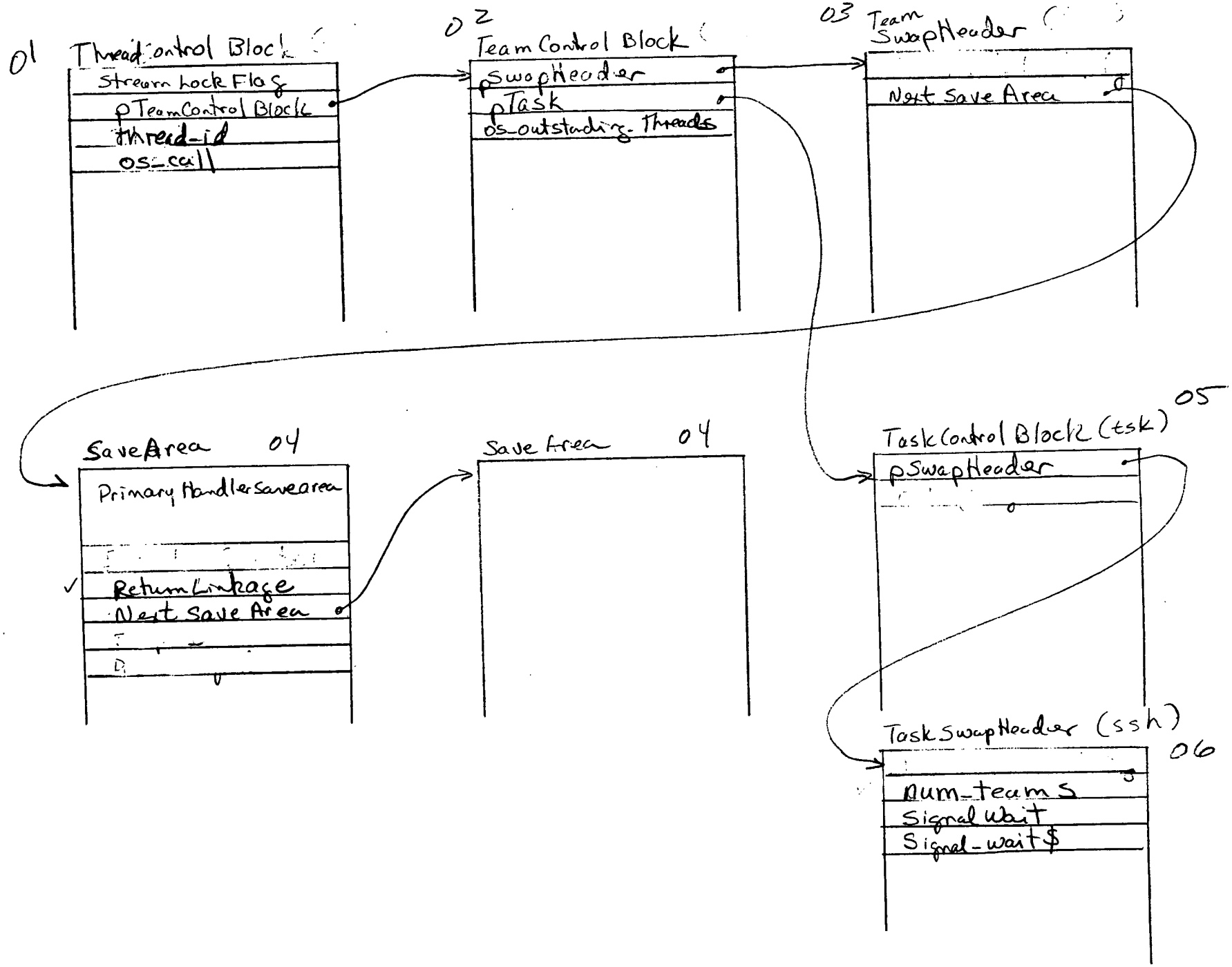

slave_return_from_swap
01
restore remaining state
02
postswap_parallel_work
03
Unlock Thread
Return

Fig 9

Figure 10


01
Thread Control Block
Stream lock Flag
pTeamControl Block
thread-id
os_call
02
Team Control Block
pSwapHeader
pTask
os_outstanding_Threads
03
Team SwapHeader
Next Save Area
05
Task Control Block (tsk)
pSwapHeader
Save Area 04
Primary Handler savearea
Return Linkage
Next Save Area
Save Area 04
Task SwapHeader (ssh)
06
num_teams
Signal wait
Signal-wait$

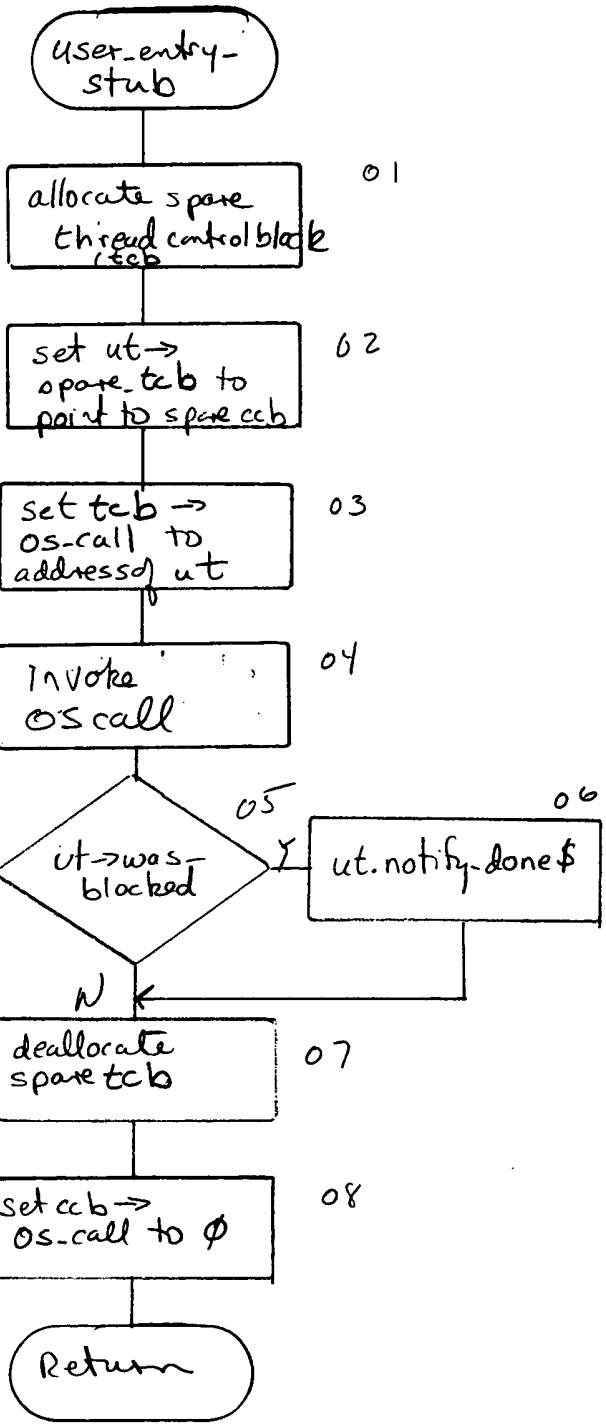

user_entry-stub
allocate spare thread control block (tcb)
01
set ut→ spare_tcb to point to spare ccb
02
set tcb→ os-call to address of ut
03
Invoke OS call
04
ut→was-blocked
05
Y
ut.notify-done$
06
N
deallocate spare tcb
07
set ccb→ os-call to ∅
08
Return

Fig 11

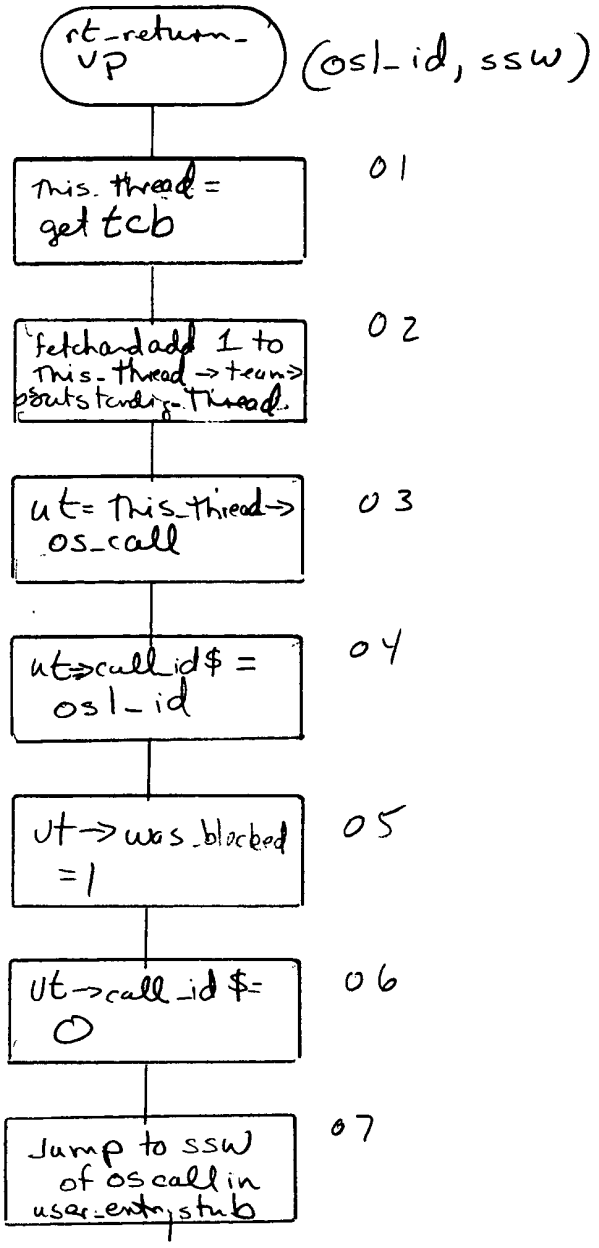

rt_return_vp
(osl_id, ssw)
this_thread = get tcb
01
fetchandadd 1 to this_thread -> team -> outstanding_thread.
02
ut = this_thread -> os_call
03
ut -> call_id$ = osl_id
04
ut -> was_blocked = 1
05
ut -> call_id $= 0
06
Jump to ssw of os call in user_entry stub
07

Fig 12

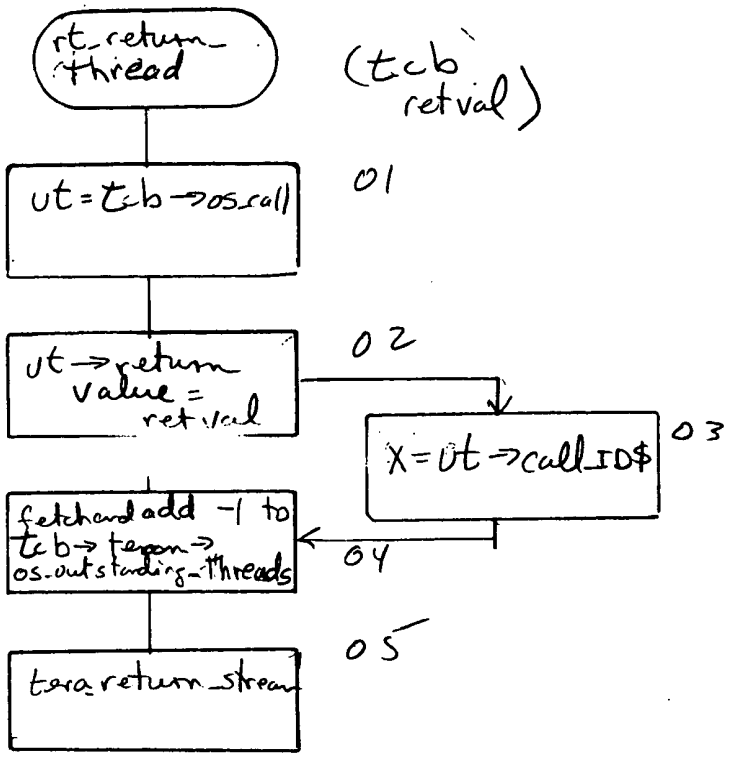

rt_return_thread
(tcb retval)
ut = tcb->os_call
01
ut->return value = retval
02
X = ut->call_ID$
03
fetchandadd -1 to tcb->team->os_outstanding_threads
04
05
tera_return_stream

Fig 13

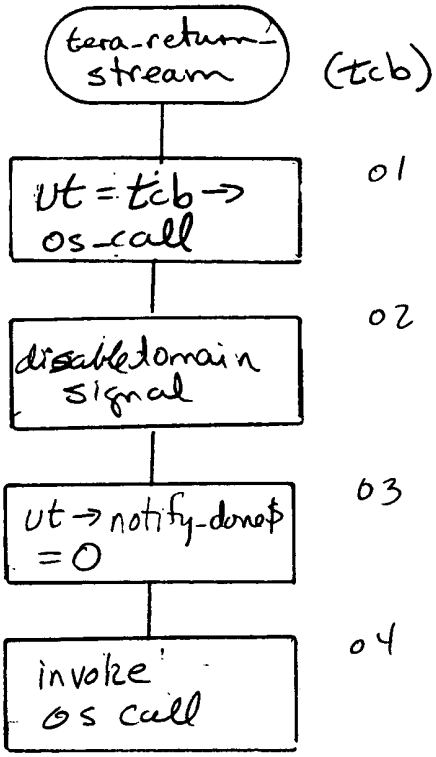

tera_return_stream
(tcb)
ut = tcb->os_call
01
disable domain signal
02
ut->notify_done$ =0
03
invoke os call
04

Fig 14

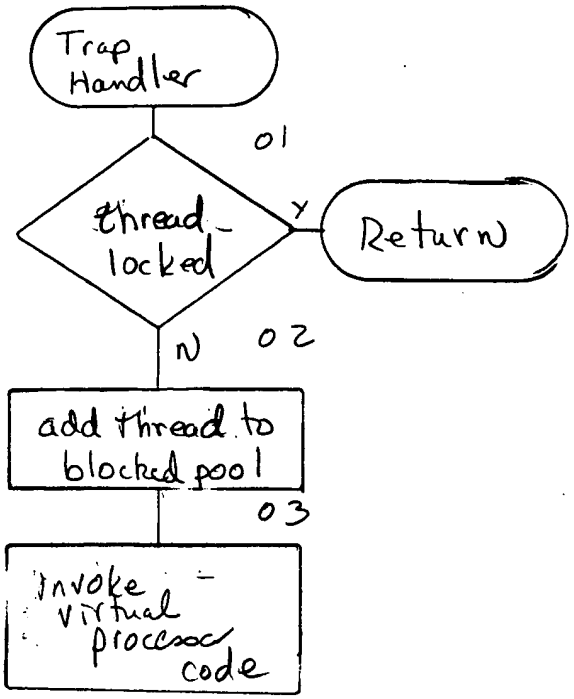

Trap Handler
01
thread_ locked
Y
Return
N
02
add thread to blocked pool
03
Invoke virtual processor code

Fig 15

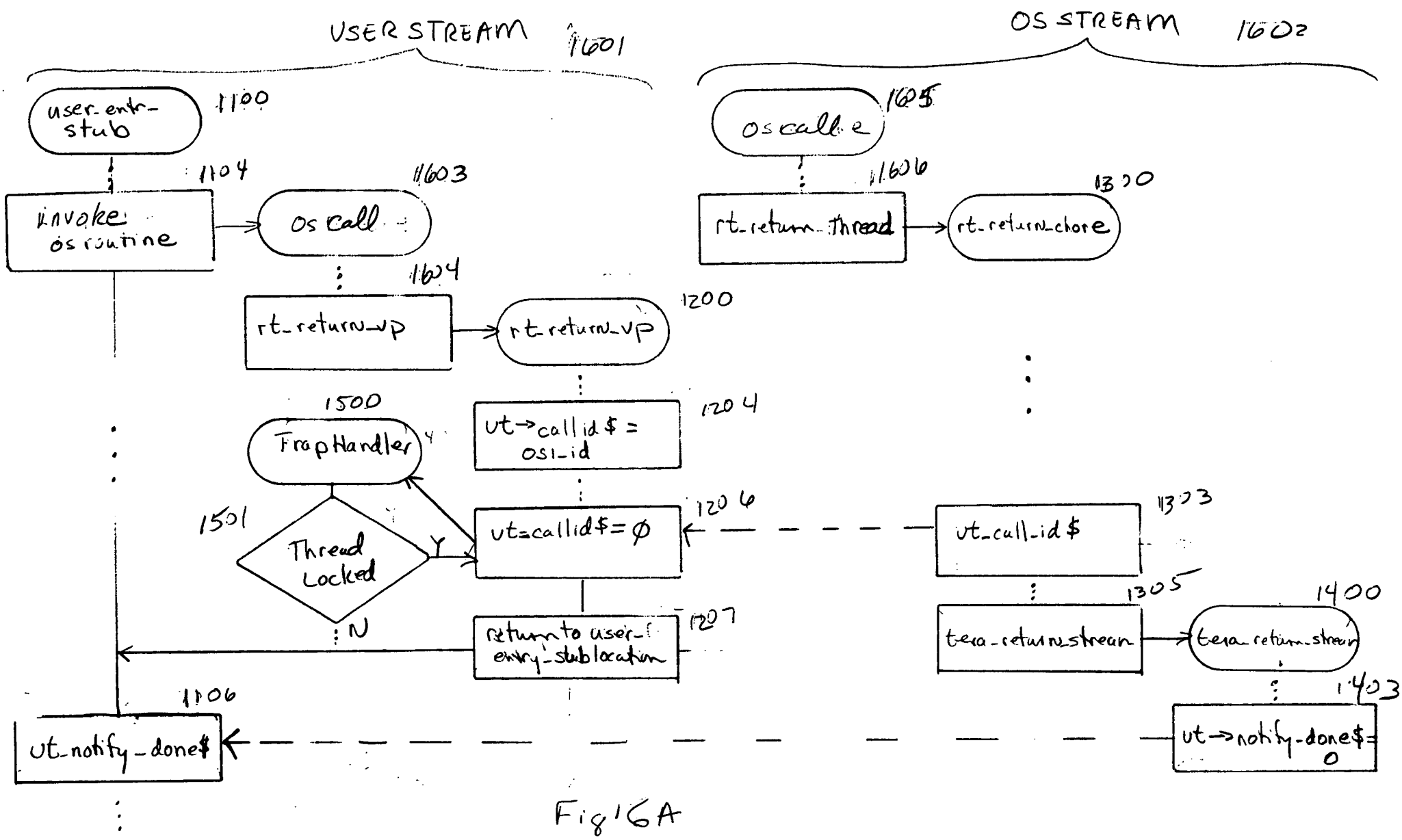

USER STREAM
1601
OS STREAM
1602
user_entr_stub
1100
oscaller
1605
1104
1603
1606
1300
invoke os routine
os call
rt_return_thread
rt_return_chore
1604
1200
rt_return_vp
rt_return_vp
1500
1204
FrapHandler
ut→callid$ = osi_id
1501
1206
1303
Thread Locked
Y
ut=callid$=∅
ut_call_id$
N
1305
1400
1207
return to user_entry_stub location
tera_return_stream
tera_return_stream
1403
1106
ut_notify_done$
ut→notify_done$=0

Fig 16A

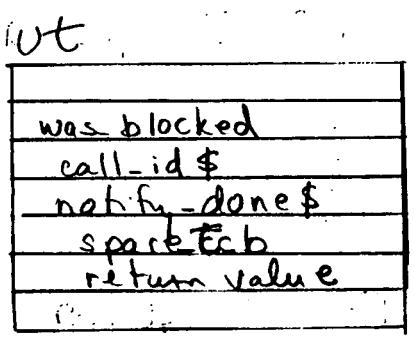



Figure 16B

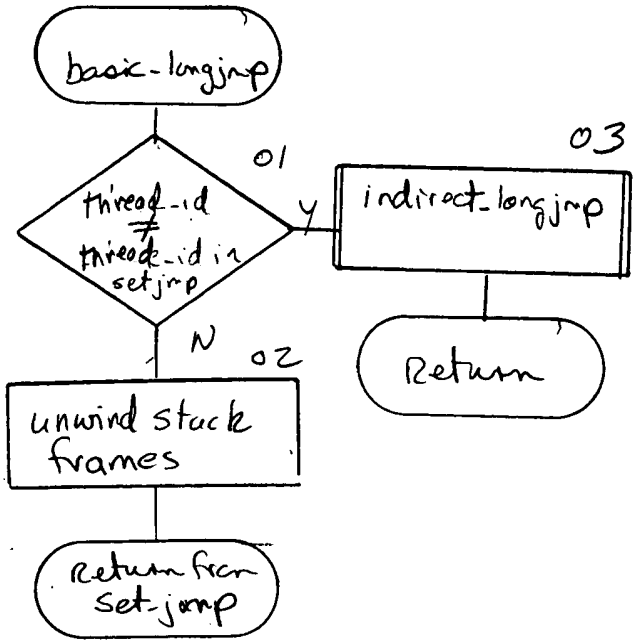

basic-longjmp
01
thread-id ≠ thread-id in setjmp
Y
03
indirect-longjmp
Return
N
02
unwind stack frames
Return from set-jmp

Fig 17

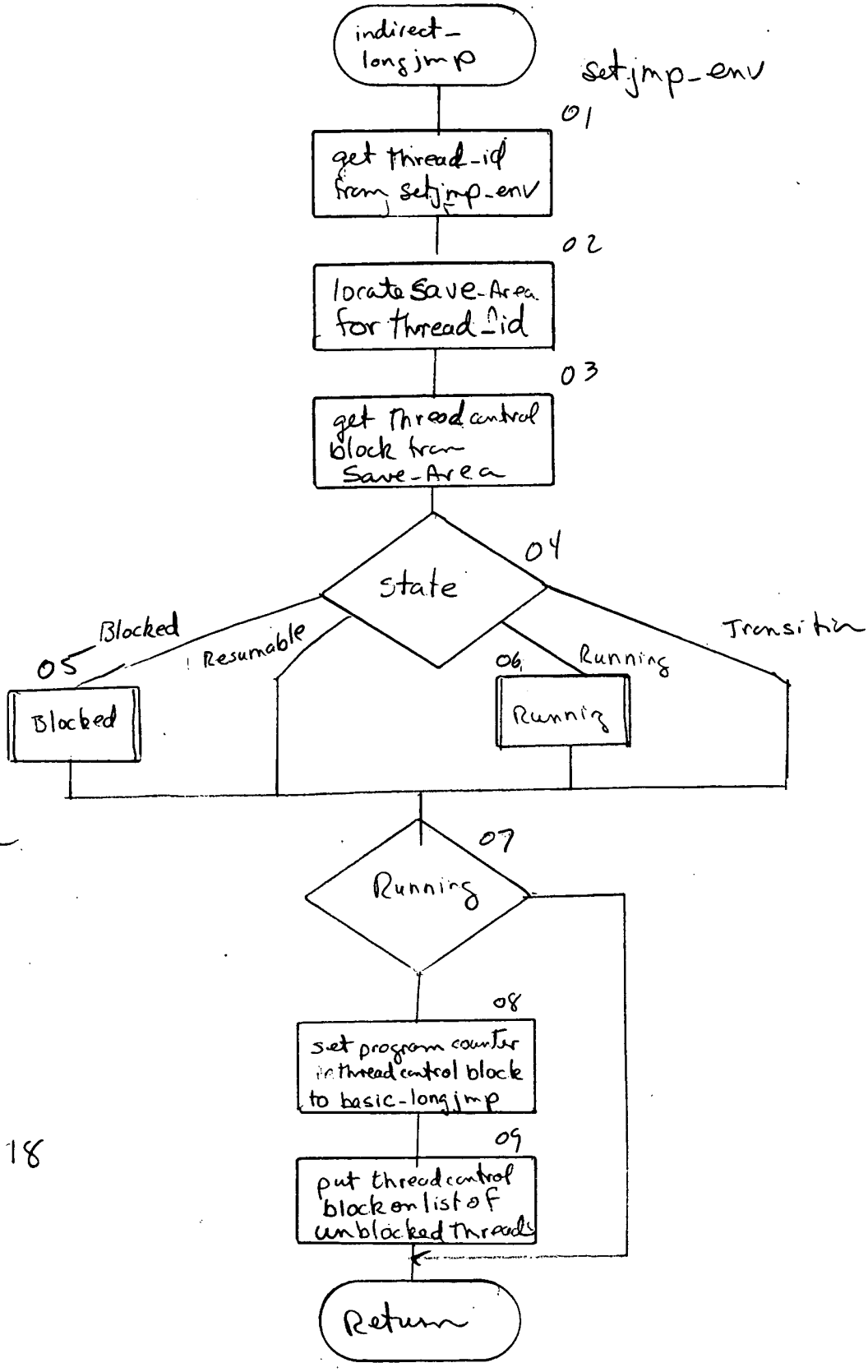

indirect_
longjmp
setjmp_env
01
get thread_id
from setjmp_env
02
locate Save-Area
for thread_id
03
get thread control
block from
Save-Area
04
State
Blocked
05
Resumable
Transition
06
Running
Blocked
Running
07
Running
08
set program counter
in thread control block
to basic-longjmp
09
put thread control
block on list of
unblocked threads
Return

Fig 18

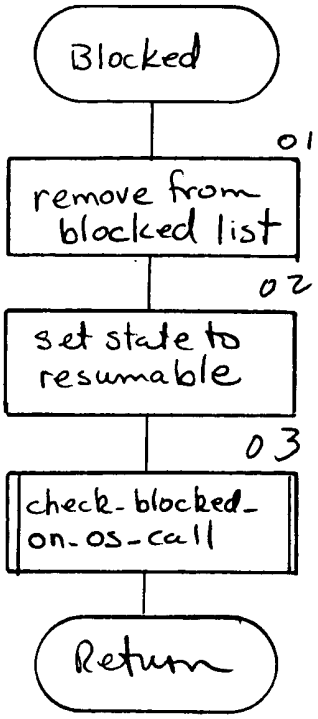

Blocked
01
remove from blocked list
02
set state to resumable
03
check-blocked-on-os-call
Return

Fig 19

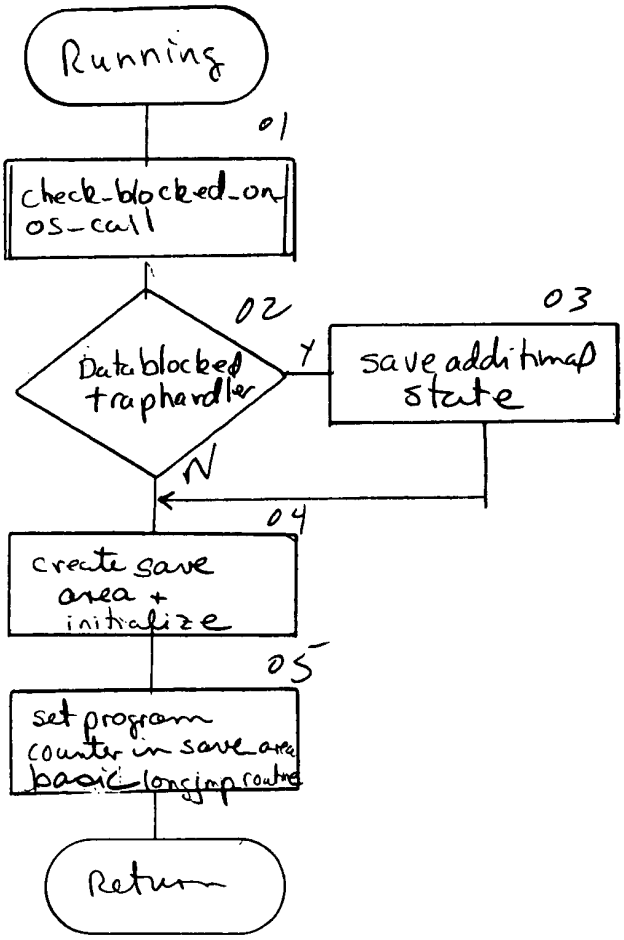
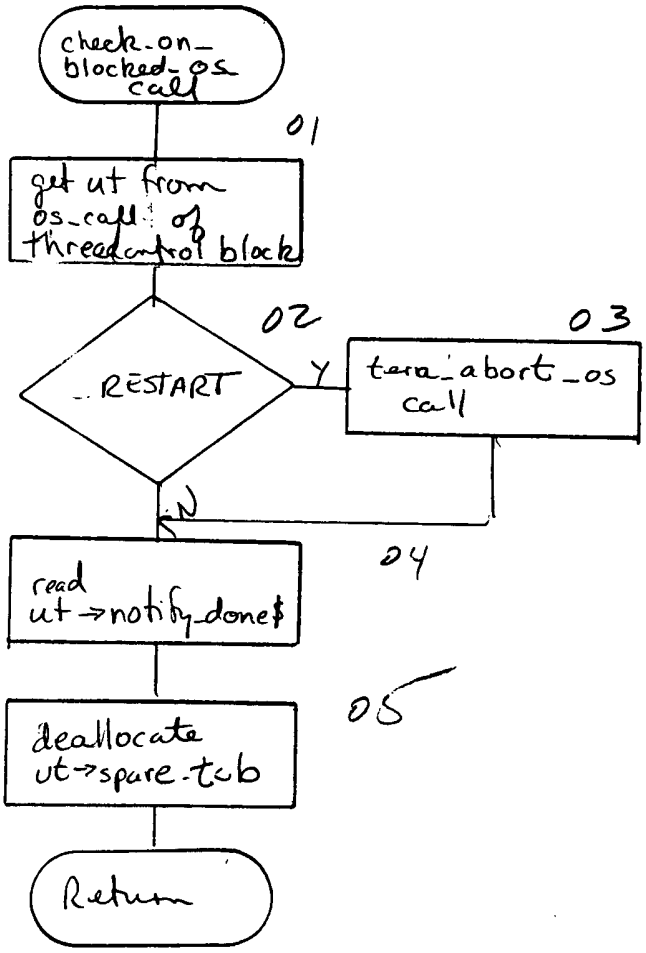

Running
01
check_blocked_on_os_call
02
Data blocked trap handler
Y
03
save addit mapstate
N
04
create save area + initialize
05
set program counter in save_area basic_longjmp routine
Return

Fig 20

Fig 21.


check_on_blocked_os_call
01
get ut from os_call of thread control block
02
RESTART
Y
03
tera_abort_os call
N
04
read ut->notify_done$
05
deallocate ut->spare_tab
Return